Bottom-Up Αντικειμενοστρεφής Προγραμματισμός

Παναγιώτης Γροντάς
Καλλιτεχνικό Γυμνάσιο με Λυκειακές τάξεις Γέρακα, pgrontas@gmail.com

Περίληψη

Στην εργασία αυτή εξετάζεται η διδακτική προσέγγιση του αντικειμενοστρεφούς προγραμματισμού στο μάθημα της Πληροφορικής στη Γ Λυκείου, η οποία οδηγεί τελικά στη διδασκαλία εννοιών αντικειμενοστρεφούς σχεδίασης και όχι προγραμματισμού. Αντ' αυτής προτείνεται μια bottom-up προσέγγιση, η οποία τεκμηριώνεται τόσο βιβλιογραφικά, όσο και από τη δομή των σχετικών προγραμμάτων σπουδών τριτοβάθμιας. Στη συνέχεια, προτείνεται μια διδακτική ακολουθία εισαγωγής του αντικειμενοστρεφούς προγραμματισμού, ανεξάρτητη γλώσσας προγραμματισμού, που αξιοποιεί τις γνώσεις και τις παραστάσεις που έχουν ήδη αποκτηθεί στο μάθημα της Πληροφορικής. Για την υποστήριξή της στην τρέχουσα μορφή του μαθήματος προτείνονται επεκτάσεις της ΓΛΩΣΣΑΣ με αντικειμενοστρεφή χαρακτηριστικά.

Λέξεις κλειδιά: αντικειμενοστρεφής προγραμματισμός, αντικειμενοστρεφής σχεδίαση, εγγραφές, αφηρημένοι τύποι δεδομένων, δομές δεδομένων.

1. Εισαγωγή

Ο αντικειμενοστρεφής προγραμματισμός (εφεξής OOP) οργανώνει τον κώδικα σε αυτόνομες μονάδες, τα αντικείμενα, που συνδυάζουν δεδομένα (ιδιότητες) και λειτουργίες (μεθόδους). Εμφανίστηκε αρχικά με γλώσσες όπως η Simula και η Smalltalk, ενώ διαδόθηκε με τη C++ και τη Java. Βασικά χαρακτηριστικά του, όπως η ενθυλάκωση, η αφαίρεση, η κληρονομικότητα και ο πολυμορφισμός, διευκολύνουν την ανάπτυξη μεγάλων συστημάτων λογισμικού, ενισχύοντας την ανεξαρτησία, τη συντηρησιμότητα και την επεκτασιμότητα του κώδικα (Armstrong, 2006). Σήμερα οι περισσότερες γλώσσες υποστηρίζουν τον OOP εγγενώς, σε συνδυασμό με ή ως επέκταση διαδικασιακών δυνατοτήτων. Η διάδοσή του οδήγησε και σε μεθοδολογίες αντικειμενοστρεφούς σχεδίασης (εφεξής OOD) και γλώσσες αναπαράστασης όπως η UML (OMG, 1997).

Η παρούσα εργασία προτείνει μια bottom-up διδακτική ακολουθία για την εισαγωγή του OOP στη δευτεροβάθμια εκπαίδευση και σχετικές ενδεικτικές επεκτάσεις για τη ΓΛΩΣΣΑ. Η ανάγκη για αυτήν τεκμηριώνεται από την κριτική θεώρηση της παρούσας κατάστασης και τη σύγκριση με την αντιμετώπιση στην τριτοβάθμια.

2. Τρέχουσα κατάσταση

Η εισαγωγή του OOP στο μάθημα της Πληροφορικής στο Γενικό Λύκειο έγινε το σχολικό έτος 2019-2020 με την έκδοση του συμπληρωματικού βιβλίου (Ινστιτούτο Εκπαιδευτικής Πολιτικής, 2019), το οποίο θα αναφέρεται εφεξής ως Β2. Εξαιτίας όμως της πανδημίας του COVID-19, ενσωματώθηκε πλήρως δύο χρόνια αργότερα,

το 2022, οπότε και εξετάστηκε στις πανελλαδικές εξετάσεις για πρώτη φορά. Στο Β2, αλλά και στις συνοδευτικές οδηγίες διδασκαλίας (Υπουργείο Παιδείας, Θρησκευμάτων και Αθλητισμού, 2025), επισημαίνεται ότι οι διδάσκοντες θα πρέπει «να αξιοποιήσουν αυθεντικά παραδείγματα» και να εκμεταλλευτούν «πρότερες γνώσεις» των μαθητών ώστε να τους επιτρέψουν να κατανοήσουν τις έννοιες του ΟΟΡ.

Οι συγγραφείς του Β2 εισάγουν τον ΟΟΡ με παραδείγματα καθημερινής ζωής, επιλογή που η βιβλιογραφία θεωρεί ανεπαρκή ειδικά όταν είναι αποκλειστική (Ragonis & Ben-Ari, 2005). Η αποστολή λουλουδιών στην Ιταλία και η ενοικίαση βιντεοκασέτας ή η παραγγελία πίτσας δύσκολα συνδέονται με αυθεντικές εμπειρίες ή πρότερες γνώσεις των μαθητών. Τα παραπάνω συμπληρώνονται με παραδείγματα κληρονομικότητας και πολυμορφισμού, κυρίως με γεωμετρικά σχήματα, όπου οι μαθητές σχεδιάζουν διαγράμματα κλάσεων και αντικειμένων χρησιμοποιώντας ένα υποσύνολό της UML. Αντίστοιχα είναι και τα θέματα των πανελλαδικών εξετάσεων από το 2022 και μετά.

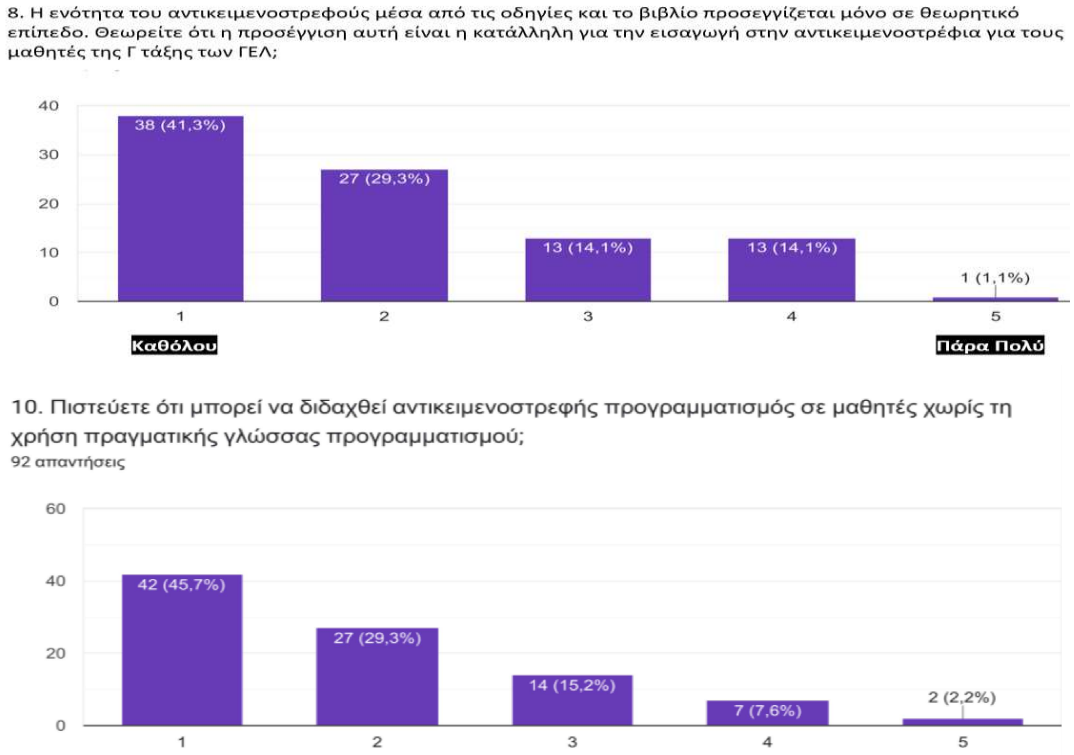
Στην προσέγγιση του Β2 οι μαθητές δεν γράφουν κώδικα, καθώς η ΓΛΩΣΣΑ δεν διαθέτει αντικειμενοστρεφείς δυνατότητες. Οι συγγραφείς δεν όρισαν δικές τους, ούτε αξιοποίησαν κάποια υπάρχουσα γλώσσα προγραμματισμού. Η δεύτερη επιλογή είναι εύλογη, επειδή θα απαιτούσε εκ νέου συγγραφή του διδακτικού πακέτου - όχι όμως και η πρώτη, αφού θα μπορούσαν να οριστούν επεκτάσεις χωρίς άμεση υλοποίηση σε προγραμματιστικό περιβάλλον. Αυτό, άλλωστε, συνέβη αρχικά και με την ίδια τη ΓΛΩΣΣΑ, που υλοποιήθηκε αργότερα και ανεξάρτητα μετά από προσωπική πρωτοβουλία καθηγητών Πληροφορικής (Γεωργόπουλος, 2021· Νικολαΐδης, 2002).

Τελικά, η εισαγωγή του ΟΟΡ φαίνεται να περιορίζεται ουσιαστικά σε ΟΟD, δηλαδή στη δημιουργία διαγραμμάτων. Η σχεδίαση προϋποθέτει εμπειρία στα χαμηλότερα επίπεδα της ταξινομίας Bloom (κατανόηση, εφαρμογή), την οποία οι μαθητές δεν έχουν αποκτήσει για αντικείμενα (Thompson et al., 2008). Επομένως, οι συγγραφείς υιοθετούν μια top-down προσέγγιση, αποκομμένη από τις μαθησιακές εμπειρίες των μαθητών στις προηγούμενες ενότητες και αντίθετη προς τις ίδιες τις συστάσεις τους. Είναι πιθανό η επιλογή αυτή να υπαγορεύτηκε κυρίως από τους περιορισμούς της ΓΛΩΣΣΑΣ και όχι από παιδαγωγικά ή επιστημονικά κριτήρια.

Η προσωπική εμπειρία του συγγραφέα στην ενεργή και συνεχή διδασκαλία της συγκεκριμένης ενότητας από το 2019 έχει δείξει ότι οι μαθητές δυσανασχετούν και παραπονιούνται για τη χρησιμότητά της και τη σχέση της με το υπόλοιπο μάθημα. Αν και τα παράπονα των μαθητών δεν αποτελούν πάντα αξιόπιστη ένδειξη, στη συγκεκριμένη περίπτωση έχουν βάση, καθώς η προσέγγιση του Β2 στον ΟΟΡ δυσαρεστεί και τους εκπαιδευτικούς (βλ. ενότητα 2.1). Η προσωπική εμπειρία, οι απόψεις κάποιων μαθητών ή εκπαιδευτικών θα μπορούσαν, αν θεωρηθούν μεμονωμένα, να απορριφθούν ως μη αξιόπιστες ενδείξεις. Αντίθετα, συνυπάρχουν ταυτόχρονα. Επιπλέον, το γεγονός ότι η προσέγγιση που ακολουθείται στην τριτοβάθμια εκπαίδευση και η βιβλιογραφία συνηγορούν υπέρ των ενδείξεων αυτών (βλ. ενότητες 2.2, 2.3) τις καθιστά μια πραγματικότητα που δεν μπορεί να αγνοηθεί.

2.1 Απόψεις εκπαιδευτικών

Η μόνη έρευνα που εντοπίστηκε για τις απόψεις ενεργών εκπαιδευτικών σχετικά με την εισαγωγή του ΟΟΡ στα ελληνικά σχολεία καταγράφεται από τις Κυρίτση και Πύρζα (2022). Βασίστηκε σε ηλεκτρονικό ερωτηματολόγιο στο οποίο απάντησαν 92 εκπαιδευτικοί ΠΕ86 με πάνω από 10 χρόνια προϋπηρεσίας. Σε αυτό, οι εκπαιδευτικοί ζητούν χρήση πραγματικής γλώσσας προγραμματισμού, κυρίως Python, εμπέδωση των εννοιών μέσω ασκήσεων και επισημαίνουν ότι η ενότητα είναι αρκετά προχωρημένη για τη συγκεκριμένη ηλικία και δεν συμβαδίζει με τη διδασκαλία του ΟΟΡ στην τριτοβάθμια. Αναφέρουν επίσης ελλείψεις σε γνώσεις πραγματικού κόσμου που απαιτούν ορισμένα παραδείγματα, όπως ο ΑΦΜ. Λόγω έλλειψης χώρου στην ενότητα αυτή αναφέρονται μόνο κάποια ενδεικτικά διαγράμματα. Ο ενδιαφερόμενος αναγνώστης παραπέμπεται στις Κυρίτση και Πύρζα (2022) για περισσότερες λεπτομέρειες.



Εικόνα 1. *Απόψεις εκπαιδευτικών για ΟΟΡ* (Κυρίτση & Πύρζα, 2022)

Παρόμοιες απόψεις διατυπώνονται και σε σχόλια ενεργών εκπαιδευτικών σε ηλεκτρονικές κοινότητες, όπως το Στέκι των Πληροφορικών, τα οποία, αν και δεν αποτελούν ερευνητικά δεδομένα, έχουν αξία (Το Στέκι των Πληροφορικών, μηνύματα 89500, 99047, 98772, 98964, 98967, 98968). Σε αυτά επισημαίνεται ότι η έλλειψη εξάσκησης δυσχεραίνει την ορθή κατανόηση των εννοιών από τους μαθητές. Επισημαίνεται επιπλέον ότι το θεωρητικό πλαίσιο δεν είναι επαρκές, δημιουργεί παρανοήσεις και τονίζεται ότι για την κατανόηση και επίλυση διαγραμματικών ασκήσεων οι μαθητές πρέπει να κάνουν υποθέσεις για τις οποίες δεν έχουν την απαιτούμενη γνώση.

2.2 Βιβλιογραφική Επισκόπηση

Για να αρθεί η υποκειμενικότητα των παραπάνω διαπιστώσεων, ερευνάται η διεθνής βιβλιογραφία. Σύμφωνα με τους Buck και Stucki (2000)[1], η πρώιμη εστίαση στη σχεδίαση αντιβαίνει στην ταξινομία του Bloom και αρχές της γνωσιακής επιστήμης. Παρ' όλα αυτά εντοπίστηκε μία εργασία που επιχείρησε να διδάξει OOD χωρίς OOP (Sims-Knight & Upchurch, 1993) σε 26 μαθητές και 8 φοιτητές. Αν και οι συγγραφείς θεωρούν ότι κάτι τέτοιο είναι δυνατό, τα αποτελέσματά τους δεν στηρίζουν επαρκώς αυτό το συμπέρασμα σε ό,τι αφορά την αποτελεσματικότητα: στην αξιολόγηση θετικά αποτελέσματα έδειξε το 54% των μαθητών έναντι 80% των φοιτητών. Σημειώνεται ότι οι μαθητές στην εργασία των Sims-Knight και Upchurch (1993) είχαν εξοικείωση με τις εφαρμογές που κλήθηκαν να σχεδιάσουν, ενώ τα αντίστοιχα θέματα του Β2 προέρχονται από τον πραγματικό κόσμο, είναι συχνά ασαφή και απαιτούν υποθέσεις χωρίς αντίστοιχη εμπειρία. Μάλιστα μία από τις πιο δημοφιλείς σχετικές μελέτες (Ragonis & Ben-Ari, 2005, σ. 209) σημειώνει[2]: *'Θα πρέπει να χρησιμοποιούνται προβλήματα σχετικά με υπολογιστικά συστήματα, και όχι μόνο προβλήματα «πραγματικής ζωής» που περιλαμβάνουν υπαλλήλους και ζώα.*'

Αυτό βέβαια δεν σημαίνει ότι πρέπει να δοθεί μόνο βάρος στον κώδικα και να αποκλειστούν εντελώς οι διαγραμματικές αναπαραστάσεις. Οι Ragonis και Ben-Ari (2005) συστήνουν ρητά τον συνδυασμό τους. Έτσι η κωδικοποίηση μπορεί να συνδυάζεται με εργαλεία διαγραμματοποίησης, είτε ενσωματωμένα στο περιβάλλον ανάπτυξης, όπως στο BlueJ είτε ξεχωριστά. Αντίστοιχη προσέγγιση ακολουθείται και σε διεθνή προγράμματα σπουδών, όπως για παράδειγμα στο πρόγραμμα σπουδών του International Baccalaureate (International Baccalaureate Organization, 2025) και στα σχετικά εγχειρίδια (Baumgarten, Ganea & Turland, 2025). Αυτό είναι απόλυτα φυσιολογικό και γίνεται άλλωστε και στον διαδικασιακό προγραμματισμό μέσω των διαγραμμάτων ροής.

2.3 Σύγκριση με προγράμματα σπουδών τμημάτων Πληροφορικής Τριτοβάθμιας

Σε πιο πρακτικό επίπεδο, η top-down προσέγγιση έρχεται σε αντίθεση με την προσέγγιση που υιοθετείται στην τριτοβάθμια εκπαίδευση. Συγκεκριμένα όπως προκύπτει από τους οδηγούς σπουδών των ελληνικών τμημάτων πληροφορικής (βλ. Παράρτημα), οι φοιτητές πρώτα έρχονται σε επαφή με τον αντικειμενοστρεφή προγραμματισμό (κατά το πρώτο / δεύτερο έτος των σπουδών τους) και αρκετά αργότερα με την αντικειμενοστρεφή σχεδίαση η οποία συναντάται για πρώτη φορά σε μαθήματα τεχνολογίας λογισμικού ή σχεδίασης πληροφοριακών συστημάτων από το 5ο εξάμηνο και μετά.

Συνοψίζοντας τα παραπάνω προκύπτουν τα εξής συμπεράσματα:

- Στα προγράμματα σπουδών της τριτοβάθμιας που εξετάστηκαν ακολουθείται η bottom-up προσέγγιση στην οποία πρώτα οι φοιτητές προγραμματίζουν σε μια

[1] Μάλιστα ονομάζουν την εργασία τους '*Design Early Considered Harmful*'.

[2] Μετάφραση του συγγραφέα.

ΟΟΡ γλώσσα και μετά σχεδιάζουν αντικειμενοστρεφή συστήματα. Αντίθετα, στη δευτεροβάθμια έχει επιλεγεί αντίστροφη πορεία, αν και απευθύνεται σε λιγότερο γνωστικά ώριμο κοινό.

- Οι καθηγητές ΠΕ86 συμφωνούν ότι η θεωρητική διδασκαλία του ΟΟΡ παρουσιάζει προβλήματα και οδηγεί σε ελλιπή κατανόηση από τους μαθητές και σε ασάφειες.
- Θα μπορούσε να προστεθεί ότι η αντικειμενοστρεφής σχεδίαση απαιτεί και γνώσεις που δεν έχουν καλλιεργηθεί στα Β1 και Β2, όπως για παράδειγμα η έννοια του πρωτεύοντος κλειδιού που προέρχεται από τις βάσεις δεδομένων.
- Μία υλοποίηση σε προγραμματιστικό περιβάλλον θα μπορούσε να αμβλύνει τα παραπάνω προβλήματα και επιπλέον θα μπορούσε να δημιουργήσει ασκήσεις στις εξετάσεις που θα μπορούσαν να κατατάξουν καλύτερα τους υποψήφιους.

3. Η bottom-up προσέγγιση

Η αντιμετώπιση των παραπάνω προβλημάτων καθιστά αναγκαία την αναθεώρηση του τρόπου διδασκαλίας του ΟΟΡ. Η εργασία εισηγείται μια bottom-up προσέγγιση, δηλαδή ξεκινάει εισάγοντας τις αντικειμενοστρεφείς έννοιες με βάση πραγματικές παραστάσεις (είτε σε μορφή ασκήσεων, είτε σε μορφή θεωρίας) που έχουν σχηματίσει οι μαθητές από την ύλη την οποία έχουν ήδη διδαχθεί. Εκμεταλλεύεται έτσι τα δύσκολα ή ασαφή σημεία, ώστε να τονίσει τα χαρακτηριστικά του ΟΟΡ. Στη συνέχεια χτίζει πάνω σε αυτά, ακολουθώντας τις αρχές του εποικοδομισμού όπως εφαρμόζεται στην Πληροφορική (Ben-Ari, 2001). Συγκεκριμένα, η διδακτική ακολουθία που προτείνεται είναι:

Παράλληλοι Πίνακες → Εγγραφές → Δομές Δεδομένων → Μέθοδοι → Ενθυλάκωση → Κληρονομικότητα → Πολυμορφισμός

Στην κατεύθυνση αυτή ορίζονται αντικειμενοστρεφείς επεκτάσεις της ΓΛΩΣΣΑΣ. Θα ήταν καλύτερα η ακολουθία να μπορούσε να εφαρμοστεί μέσω μιας πραγματικής γλώσσας προγραμματισμού, αλλά δεν υπάρχουν στοιχεία για το αν και πότε κάτι τέτοιο θα καταστεί εφικτό.

3.1 Αφαίρεση δεδομένων μέσω εγγραφών

Οι μαθητές έρχονται για πρώτη φορά σε επαφή με την έννοια της αφαίρεσης μέσω της ενότητας των υποπρογραμμάτων, με τη βοήθεια των οποίων ομαδοποιούν εντολές και αναφέρονται σε αυτές συνολικά με ένα όνομα. Δημιουργούν έτσι νέες εντολές οι οποίες αποκρύπτουν τις λεπτομέρειες υλοποίησής τους, με τον ίδιο τρόπο που κατά τη χρήση της εντολής Διάβασε π.χ. κρύβονται οι λεπτομέρειες μετακίνησης των bits από το πληκτρολόγιο στη μνήμη.

Η αφαίρεση δεδομένων αναφέρεται θεωρητικά στο Β1 (ορισμός δομής δεδομένων, εγγραφή), αλλά δεν υλοποιείται προγραμματιστικά. Αυτό έχει ως αποτέλεσμα να χρησιμοποιούνται ‘άκομψες’ τεχνικές όπως συμβαίνει με τους ‘παράλληλους’ πίνακες, όπου η συσχέτιση δεδομένων της ίδιας οντότητας γίνεται με το να έχουν τον ίδιο δείκτη σε διαφορετικούς πίνακες. Οι μαθητές λοιπόν θα πρέπει να προσέχουν

όταν διαταράσσουν αυτή τη σχέση (π.χ. σε κάποια ταξινόμηση) να προβαίνουν και σε ενέργειες αποκατάστασής της. Η κατάσταση αυτή αποτελεί το αντίθετο της αφαίρεσης, αφού αναγκάζει την ενασχόληση με λεπτομέρειες που δεν σχετίζονται με το πρόβλημα που επιλύεται.

Η εισαγωγή της αφαίρεσης δεδομένων μπορεί να γίνει με την προσθήκη δυνατότητας δημιουργίας σύνθετων τύπων δεδομένων στη ΓΛΩΣΣΑ. Για τον σκοπό αυτό προτείνεται η προσθήκη μιας νέας εντολής **ΤΥΠΟΣ**.

```
Τύπος Μαθητής
Ιδιότητες:
 Χαρακτήρες: Όνομα, Επώνυμο
 Ακέραιες: Απουσίες
 Πραγματικές: Βαθμός
Τέλος Μαθητής
```

Η δήλωση μεταβλητών μπορεί να γίνει στο τμήμα δηλώσεων του προγράμματος, επεκτείνοντας τους ενσωματωμένους τύπους, ως εξής: **Μαθητής: ΓΟικΠλη[50]**. Τονίζεται ότι οι σύνθετοι τύποι δεδομένων δηλώνονται σε ενικό αριθμό, ενώ οι ενσωματωμένοι σε πληθυντικό. Αυτό επηρεάζει μεν την αναγνωσιμότητα του προγράμματος, αλλά είναι συνέπεια του τρόπου με τον οποίο οι δημιουργοί της ΓΛΩΣΣΑΣ επέλεξαν να συντάξουν το τμήμα δηλώσεων. Στις περισσότερες γλώσσες προγραμματισμού, οι τύποι των μεταβλητών αναγράφονται στον ενικό κατά τη δήλωσή τους π.χ. (Free Pascal Documentation, 2025· Microsoft, 2021).

Όπως και στις πραγματικές γλώσσες προγραμματισμού, για να υπάρξει αναφορά σε συγκεκριμένη ιδιότητα, μπορεί να χρησιμοποιηθεί ο συμβολισμός της τελείας (dot notation). Για παράδειγμα, η έκφραση **ΓΟικΠλη[1].Όνομα** απομονώνει το πεδίο **όνομα** του πρώτου μαθητή. Παρακάτω συγκρίνεται η εισαγωγή δεδομένων με τον υπάρχοντα και με τον προτεινόμενο τρόπο:

```
Για μαθ από 1 μέχρι 50
   Διαβασε Όνομα[μαθ]
   Διαβασε Επώνυμο[μαθ]
   Διαβασε Βαθμός[μαθ]
   Διάβασε Απουσίες[μαθ]
Τέλος_Επανάληψης
```

```
Για μαθ από 1 μέχρι 50
   ΓΟικΠλη[μαθ] ← Δημιούργησε Μαθητής
   Διαβασε ΓΟικΠλη[μαθ].Όνομα
   Διαβασε ΓΟικΠλη[μαθ].Επώνυμο
   Διαβασε ΓΟικΠλη[μαθ].Βαθμός
   Διαβασε ΓΟικΠλη[μαθ].Απουσίες
Τέλος_Επανάληψης
```

Αξίζει να επισημανθεί ότι προτείνεται η χρήση της λέξης κλειδιού **ΤΥΠΟΣ** αντί για αντίστοιχες λέξεις όπως αυτές που υπάρχουν σε άλλες γλώσσες (ενδεικτικά εγγραφή ή δομή) για να τονιστεί στους μαθητές ότι πρόκειται για τύπους δεδομένων – όπως οι ενσωματωμένοι – με τη διαφορά ότι μπορούν να περιέχουν σύνθετα δεδομένα και ότι μπορούν να οριστούν από τους ίδιους. Η διαφορά αυτή εκφράζεται στον κώδικα με τη χρήση της νέας λέξης κλειδιού **Δημιούργησε**. Στην πρώτη επαφή με τους σύνθετους τύπους δεδομένων δεν χρειάζεται, και καλό θα ήταν να μη γίνει,

αναφορά σε κλάσεις και αντικείμενα. Προτείνεται η εισαγωγή τους να γίνει μέσω των Δομών Δεδομένων.

3.2 Προσθήκη Μεθόδων – Ενθυλάκωση

Το επόμενο βήμα στην bottom-up προσέγγιση είναι η προσθήκη μεθόδων στους σύνθετους τύπους δεδομένων, φτάνοντας έτσι στον ορισμό της κλάσης. Για τον σκοπό αυτό προστίθεται η λέξη κλειδί **Μέθοδοι**. Για τον ορισμό των μεθόδων επαναχρησιμοποιείται η γνωστή σύνταξη από τα υποπρογράμματα.

Η εισαγωγή του αντικειμενοστρεφούς προγραμματισμού συνδέεται με τη διδασκαλία των δομών δεδομένων, καθώς συνδυάζουν δεδομένα και λειτουργίες. Για παράδειγμα, η στοίβα μπορεί να δηλωθεί ως εξής:

```
Τύπος Στοίβα
Ιδιότητες:
 Ακέραιες: Α[100], Κορυφή, Χωρητικότητα
Μέθοδοι:
 Διαδικασία Ώθηση(Στοιχείο)
 Διαδικασία Απώθηση(Στοιχείο)
 Συνάρτηση Είναι_Άδεια():Λογική
Τέλος Στοίβα
```

Όπως και νωρίτερα, για τη δημιουργία αντικειμένων μπορεί να χρησιμοποιηθεί η εντολή **Δημιούργησε**. Για λόγους απλότητας, η αρχικοποίηση των ιδιοτήτων μπορεί να μη γίνεται στον κατασκευαστή, αλλά μέσω μεθόδων ορισμού τιμών (setter).

Για να επιτευχθεί η ενθυλάκωση, εισάγονται καθοριστές πρόσβασης (access modifiers) χρησιμοποιώντας λέξεις κλειδιά όπως **Δημόσια** και **Ιδιωτική**, οι οποίοι θα συνοδεύουν κάθε μεταβλητή και μέθοδο. Η προκαθορισμένη επιλογή θα είναι η δημόσια πρόσβαση, ώστε να υπάρξει εύκολη χρήση των κλάσεων ως σύνθετοι τύποι δεδομένων. Η ύπαρξη όμως δυνατότητας ιδιωτικής πρόσβασης επιτρέπει την πρακτική υλοποίηση της ενθυλάκωσης. Με αυτές είναι δυνατό, για παράδειγμα, οι ιδιότητες του τύπου της Στοίβας να είναι ιδιωτικές και κατά συνέπεια να μην μπορεί να υπάρχει εξωτερική πρόσβαση στον πίνακα που φιλοξενεί τα στοιχεία της στοίβας, κάτι που με την υλοποίηση των Β1 και Β2 επιτρέπεται.

3.3 Κληρονομικότητα και Πολυμορφισμός

Για την υλοποίηση της κληρονομικότητας ορίζεται η λέξη κλειδί **Επεκτείνει** η οποία μπορεί να χρησιμοποιηθεί για να οριστεί μια υποκλάση. Στον ορισμό της υποκλάσης παραλείπονται οι ιδιότητες / μέθοδοι που κληρονομούνται και ορίζονται μόνο οι καινούριες ή αυτές που παρακάμπτονται (override). Με αυτό τον τρόπο υπάρχει συμβατότητα με το πώς σχεδιάζονται τα (ιεραρχικά) διαγράμματα κλάσεων στην υπάρχουσα προσέγγιση του Β2.

```
Τύπος Μαθητής_Λυκείου Επε-
κτείνει Μαθητής
Ιδιότητες:
Μέθοδοι:
 Συνάρτηση Είναι_Άριστα():Λο-
γική
Τέλος Μαθητής_Λυκείου
```

```
Τύπος Μαθητής_Γυμνασίου Επε-
κτείνει Μαθητής
Ιδιότητες:
Μέθοδοι:
      Συνάρτηση Είναι_Άρι-
στα():Λογική
Τέλος Μαθητής_Γυμνασίου
```

Στο παραπάνω, υποτίθεται ότι στον τύπο Μαθητής έχει προστεθεί η μέθοδος Είναι_Άριστα() την οποία οι υποκλάσεις παρακάμπτουν. Με βάση την κληρονομικότητα ο πολυμορφισμός υλοποιείται εύκολα:

```
ΜΕΤΑΒΛΗΤΕΣ
  Μαθητής: Μ[2]
Αρχή
 Μ[1]  ← Δημιούργησε Μαθητής_Λυκείου
 Μ[1].Βαθμός ← 18.2
 Μ[2] ← Δημιούργησε Μαθητής_Γυμνασίου
 Μ[2].Βαθμός ← 18.2
 Για μαθ από 1 μέχρι 2
   ΓΡΑΨΕ Μ[μαθ].Είναι_Άριστα()
 Τέλος_Επανάληψης
```

Στο παραπάνω τμήμα κώδικα, τόσο το αντικείμενο στη θέση **Μ[1]**, όσο και το αντικείμενο στη θέση **Μ[2]** είναι τύπου **Μαθητής** αφού τόσο οι μαθητές λυκείου όσο και οι μαθητές γυμνασίου είναι υποσύνολα του συνόλου των μαθητών. Κατά συνέπεια μπορούν να τοποθετηθούν σε πίνακα με τύπο το υπερσύνολο. Αρχικοποιώντας τους μαθητές με τον ίδιο βαθμό και καλώντας τη μέθοδο Είναι_Άριστα() προκύπτουν διαφορετικά αποτελέσματα, καθώς το άριστα για τους μαθητές του Λυκείου είναι από 18,1 και πάνω, ενώ για τους μαθητές Γυμνασίου από 18,5 και πάνω. Με το παράδειγμα αυτό μπορεί να αναδειχθεί η χρησιμότητα και η εφαρμογή του πολυμορφισμού.

4. Συμπεράσματα

Η εργασία αυτή είχε ως αφετηρία τη δυσαρέσκεια των μαθητών που παρατηρεί ο συγγραφέας στη διδακτική πράξη, η οποία, όπως αναλύθηκε, υποστηρίζεται από τις απόψεις των διδασκόντων, τη βιβλιογραφία αλλά και την πρακτική στην τριτοβάθμια. Επιπλέον κρίνεται εύλογη, καθώς τους ζητείται να κινηθούν στο υψηλότερο επίπεδο αφαίρεσης της OOD χωρίς προηγούμενη εμπειρία στο επίπεδο της υλοποίησης. Η προσέγγιση αυτή αντιστρέφει την πρακτική των περισσότερων προγραμμάτων σπουδών, όπου ο προγραμματισμός θεμελιώνει και κινητοποιεί τη σχεδίαση. Η εργασία προτείνει μια bottom-up διδασκαλία που επιτρέπει στους μαθητές να πειραματιστούν και να αξιοποιήσουν τις ήδη αποκτημένες γνώσεις τους. Αν και η

προτεινόμενη προσέγγιση είναι ανεξάρτητη γλώσσας προγραμματισμού, προτείνονται επεκτάσεις στη ΓΛΩΣΣΑ για την καλύτερη σύνδεσή της με την υπάρχουσα εκπαιδευτική πραγματικότητα. Η υλοποίηση των επεκτάσεων αυτών σε προγραμματιστικό περιβάλλον, καθώς και η πειραματική αξιολόγηση της πρότασης μέσω αποτίμησης και σύγκρισης των μαθησιακών αποτελεσμάτων υπερβαίνουν τα όρια της παρούσας εργασίας και θα εξεταστούν σε μελλοντική δουλειά.

Ευχαριστίες. Ευχαριστούμε την κ. Φανή Πύρζα για την πρόσβαση που μας έδωσε στα στοιχεία της (Κυρίτση & Πύρζα, 2022).

Αναφορές

Armstrong, D. J. (2006). The quarks of object-oriented development. *Communications of the ACM*, *49*(2), 123-128.

Baumgarten, P., Ganea, I., & Turland, C. (2025). *Computer Science for the IB Diploma*Hodder Education / Hachette Learning.

Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching,* 20(1), 45-73. AACE.

Buck, D., & Stucki, D. J. (2000). *Design early considered harmful: Graduated exposure to complexity and structure based on levels of cognitive development*. In Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education (pp. 75–79). ACM. https://doi.org/10.1145/330908.331817

Free Pascal Documentation. (2025). *Variable Declaration*. https://www.freepascal.org/docs-html/ref/refse22.html

International Baccalaureate Organization. (2025). *Computer science guide: Diploma Programme (First assessment 2027).* International Baccalaureate Organization.

Microsoft. (2021). *Data type summary – Visual Basic*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/visual-basic/language-reference/data-types

Object Management Group. (1997). *Unified Modeling Language (OMG UML), version 1.1*. https://www.omg.org/spec/UML/1.1

Ragonis, N., & Ben-Ari, M. (2005). A long-term investigation of the comprehension of OOP concepts by novices. *Computer Science Education, 15*(3), 203–221.

Sims-Knight, J. E., & Upchurch, R. L. (1993). *Teaching object-oriented design without programming: A progress report. Computer Science Education, 4*(1), 135–156.

Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. (2008). *Bloom's taxonomy for CS assessment*. In S. Hamilton & M. Hamilton (Eds.), ACE '08: Proceedings of the tenth conference on Australasian computing education – Volume 78 (pp. 155–161). Australian Computer Society Inc.

Γεωργόπουλος, Α. (2021). *Ανάπτυξη, υποστήριξη και μελέτη ολοκληρωμένου προγραμματιστικού περιβάλλοντος και διαδικτυακής κοινότητας πρακτικής για τη*

διδασκαλία της Πληροφορικής στο Γενικό Λύκειο [Διδακτορική διατριβή, Πανεπιστήμιο Πατρών]. Εθνικό Αρχείο Διδακτορικών Διατριβών. https://doi.org/10.12681/eadd/52285

Ινστιτούτο Εκπαιδευτικής Πολιτικής. (2019). *Πληροφορική: Βιβλίο μαθητή – Συμπληρωματικό εκπαιδευτικό υλικό (Γ' ΓΕΛ, Ομάδα Προσανατολισμού Σπουδών Οικονομίας & Πληροφορικής)*. Ινστιτούτο Τεχνολογίας Υπολογιστών και Εκδόσεων «Διόφαντος».

Κυρίτση, Α., & Πύρζα, Φ. (2022, 24 Οκτωβρίου). *Η Πληροφορική στο Γυμνάσιο και το Γενικό Λύκειο* [Ομιλία σε διαδικτυακή ημερίδα]. Στο *Πληροφορική και Πληροφορικοί, σε όλους τους τύπους σχολείων πρωτοβάθμιας και δευτεροβάθμιας*. Ομάδα Τεχνικής Υποστήριξης 4ου ΠΕΚΕΣ Αττικής. YouTube. https://www.youtube.com/watch?v=KIOASDVGlcc

Νικολαΐδης, Σ. (2002). *ΓλωσσοΜάθεια* [Λογισμικό]. https://spinet.gr/glossomatheia/

Το Στέκι των Πληροφορικών (2002) [κοινότητα πρακτικής και μάθησης]. https://al-kisg.mysch.gr/steki/index.php.

Υπουργείο Παιδείας, Θρησκευμάτων και Αθλητισμού. (2025). *Οδηγίες διδασκαλίας Πληροφορικής προσανατολισμού Γ΄ τάξης Γενικού Λυκείου για το σχ. έτος 2025-2026*.

Παράρτημα. Αντικειμενοστρεφής προγραμματισμός στην Τριτοβάθμια

Πίνακας 1. *Διδασκαλία OOP και OOD σε τμήματα πληροφορικής*

Τμήμα	Εξάμηνο OOP	Εξάμηνο OOD / Μάθημα
Τμήμα Πληροφορικής, Οικονομικό Πανεπιστήμιο Αθηνών https://www.dept.aueb.gr/el/cs/courses	2ο	5ο / Τεχνολογία Λογισμικού
Τμήμα Πληροφορικής, Πανεπιστήμιο Πειραιά https://cs.unipi.gr/proptyxiakes-spoudes/	2ο	6ο / Τεχνολογία Λογισμικού
Τμήμα Πληροφορικής, ΕΚΠΑ https://www.di.uoa.gr/studies/undergraduate/courses/	3ο	6ο εξάμηνο / Ανάλυση/Σχεδίαση Συστημάτων Λογισμικού
Τμήμα Πληροφορικής και Τηλεματικής, Χαροκόπειο Πανεπιστήμιο Αθηνών https://dit.hua.gr/index.php/el/programmata-spoudon/proptyxiako/mathimata	2ο	7ο / Πληροφοριακά Συστήματα

Τμήμα Πληροφορικής ΑΠΘ https://csd.auth.gr/studies/undergraduate/courses/	3ο	5ο / Τεχνολογία Λογισμικού
Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης https://www.csd.uoc.gr/CSD/uploaded_files/CSD_OdhgosSpoudwn_PPS_2025-26_epikairopoihsh11-2025.pdf	3ο	3ο / Αντικειμενοστρεφής προγραμματισμός και κυρίως 6ο / Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων
Τμήμα Πληροφορικής Πανεπιστημίου Δ. Μακεδονίας https://cs.uowm.gr/proptychiako-programma-spoydon/	2ο	5ο / Τεχνολογία Λογισμικού
Τμήμα Πληροφορικής Πανεπιστημίου Θεσσαλίας (https://dit.uth.gr/?page_id=36374)	2ο	6ο / Τεχνολογία Λογισμικού

Bottom-Up Object-Oriented Programming

Panagiotis Grontas
Arts School of Gerakas, pgrontas@gmail.com

Abstract

This work examines the approach to teaching object-oriented programming in the Greek Computer Science curriculum for the final year of high school. The examination reveals that the current top-down methodology concentrates predominantly on object-oriented design rather than programming. To address this issue, a bottom-up approach is proposed —a paradigm well-supported by academic literature and aligned with university-level computer science curricula. Furthermore, the paper presents a language-agnostic framework for introducing object-oriented programming, designed to build directly upon the foundational knowledge and technical skills already established in the current curriculum. To facilitate seamless integration within the existing course structure, extensions to GLOSSA (ΓΛΩΣΣΑ) featuring object-oriented capabilities are introduced.

Keywords: object-oriented programming, object-oriented design, records, abstract data types, data structures